

# Creating Executable Programs on the mainframe

## **A general procedure for developing a COBOL program on a mainframe**

1. Use ISPF to enter the source code for the source program into a file.
2. Use ISPF to submit the JCL for running a compile-link-and-go procedure.
3. If necessary, correct any compile-time errors and submit the JCL for the compile-link-and-go procedure again. Repeat this step until all compile-time errors are corrected.
4. If necessary, fix any run-time errors that occur when the program is executed and submit the JCL for the compile-link-and-go procedure again. Repeat this step until all run-time errors are corrected.
5. Use SDSF or other facilities to review the test run output to make sure the program works correctly.

# A compile-link-and-go procedure on a mainframe consists of three JCL steps

## Step 1

- The COBOL compiler compiles the *source program* into an *object module*, incorporating any copy members it needs.
- The compiler produces output that can be printed or displayed.

## Step 2

- The *linkage editor* links the object program with any subprograms it needs, thus creating an executable program called a *load module*.

## Step 3

- The executable program runs so you can see whether it works correctly.

# A JCL Overview

## JCL statements for a job that prints a library member

Identifier field

Name field

Operation field

Parameters field

//MM01P	JOB	(36512) , 'A PRINCE' , NOTIFY=&SYSUID
//STEP1	EXEC	PGM=IEBGENER
//SYSPRINT	DD	SYSOUT=*
//SYSUT1	DD	DSN=MM01.COPYLIB.COBOL (CUSTMAST) , DISP=SHR
//SYSUT2	DD	SYSOUT=*
//SYSIN	DD	DUMMY

# An introduction to jobs and Job Control Language

- A job consists of one or more *job steps*. Each job step executes a program or procedure.
- The *identifier field* identifies the statement as a JCL statement. For most statements, it consists of two slashes (//) in columns 1 and 2.
- The *name field* associates a name with the statement. The name consists of one to eight characters and must begin in column 3.
- The *operation field* specifies the statement's function. It can be coded in any column, as long as it's separated from the name field by at least one blank.
- The *parameters field* begins at least one position after the end of the operation field and can extend into column 71. It can contain one or more *parameters* that affect how the statement is processed.
- To continue a statement, break the parameter field after a comma, code slashes in columns 1 and 2 of the following line, and code the next parameter beginning anywhere in columns 4 through 16.

# The “JOB” Statement

The JOB statement must be the first statement in a job. It supplies a *job name* to identify the job, along with accounting information and various options that affect how the job is processed.

```
//MM01P      JOB    (36512) , 'A PRINCE' , NOTIFY=&SYSUID  
//STEP1      EXEC   PGM=IEBGENER  
//SYSPRINT   DD     SYSOUT=*  
//SYSUT1     DD     DSN=MM01.COPYLIB.COBOL(CUSTMAST) , DISP=SHR  
//SYSUT2     DD     SYSOUT=*  
//SYSIN      DD     DUMMY
```

# The “EXEC” Statement

The EXEC statement identifies the program or procedure to be executed in the job step. It can have a *step name* in the name field.

```
//MM01P      JOB    (36512) , 'A PRINCE' , NOTIFY=&SYSUID  
//STEP1      EXEC   PGM=IEBGENER  
//SYSPRINT   DD     SYSOUT=*  
//SYSUT1     DD     DSN=MM01.COPYLIB.COBOL (CUSTMAST) , DISP=SHR  
//SYSUT2     DD     SYSOUT=*  
//SYSIN      DD     DUMMY
```



# The “DD” Statement

- The DD statement allocates the data sets required by the program or procedure. The *ddname* in the name field must be the same as the one that’s used in the program or procedure.
- You typically code the DD statement for a printer file using the SYSOUT = \* format. Then, the output is processed based on the default output class or the output class you specify in the MSGCLASS parameter of the JOB statement.
- You can specify a dummy data set by coding the DUMMY parameter on its DD statement.

```
//MM01P      JOB   (36512) , 'A PRINCE' , NOTIFY=&SYSUID  
//STEP1      EXEC  PGM=IEBGENER  
//SYSPRINT  DD     SYSOUT=*  
//SYSUT1     DD     DSN=MM01.COPYLIB.COBOL(CUSTMAST) , DISP=SHR  
//SYSUT2     DD     SYSOUT=*  
//SYSIN      DD     DUMMY
```

## The cataloged procedures for COBOL program development

- A *cataloged procedure* is a pre-written segment of JCL code.
- To use a cataloged procedure, you invoke it using JCL. Within that JCL, you identify the required data sets using the ddnames that are in the procedure.
- The COBOL compile-and-go procedures use the *loader* rather than the linkage editor. The loader is similar in function to the linkage editor, but it doesn't create a load module.
- If you use a compile-and-go procedure, you include the data sets that are normally used for the LKED step on the GO step instead.

## Cataloged procedures for COBOL program development

Procedure	VS COBOL II	COBOL for MVS, OS/390, or z/OS
Compile only	COB2UC	IGYWC
Compile and link	COB2UCL	IGYWCL
Compile, link, and go	COB2UCLG	IGYWCLG
Compile and go	COB2UCG	IGYWCG

## Step names used in the cataloged procedures

Step	VS COBOL II	COBOL for MVS, OS/390, or z/OS
Compile	COB2	COBOL
Link	LKED	LKED
Go	GO	GO

## How to use the compile-link-and-go procedure for COBOL for z/OS

- *Compiler options* are used to control the compiler's execution and output. To change the default options in the COBOL step, code the PARM.COBOL parameter on the EXEC statement. Then, include the compiler options you want to use in quotes.
- When you execute a procedure, the ddnames should include the appropriate step name in the procedure.
- If the program requires input or output files, you must code DD statements in the GO step to identify those files. The names you use on the DD statements must be the same as the ddnames in the system names of the Select statements used in the program.
- You can use the ISPF editor to create JCL jobs. Then, you can submit the job for processing by issuing the SUBMIT primary command from the ISPF edit data display.

## DD statements used with cataloged procedures

Step	ddname	Description
COB2/COBOL	SYSIN	Source program input for the COBOL compiler.
	SYSLIB	A library that's searched for copy members.
	SYSLIN	Object module output.
LKED	SYSLIB	Subprogram library.
	SYSLIN	Object module input.
	SYSIN	Additional object module input.
	SYSLMOD	Load module output.
GO	SYSOUT	Output from DISPLAY statements.
	SYSIN	Input for ACCEPT statements.
	SYSDBOUT	Symbolic debugging output.
	SYSUDUMP	Abnormal termination dump output.
	SYSABEND	

# JCL that invokes the compile-link-and-go procedure for COBOL for z/OS

```
//MM01CLG JOB (36512), 'R MENENDEZ', MSGLEVEL=(1,1), REGION=4M,  
//          MSGCLASS=X, CLASS=A, NOTIFY=&SYSUID  
//*-----*  
//*  COMPILE, LINK, AND EXECUTE A COBOL FOR Z/OS AND OS/390 PROGRAM  
//*-----*  
//STEP1    EXEC PROC=IGYWCLG, PARM.COBOL='XREF, FLAG(I, E) '  
//COBOL.SYSIN DD DSN=MM01.TEST.COBOL(RPT1000), DISP=SHR  
//COBOL.SYSLIB DD DSN=MM01.TEST.COPYLIB, DISP=SHR  
//*-----*  
//LKED.SYSLMOD DD DSN=MM01.TEST.LOADLIB(RPT1000), DISP=SHR  
//LKED.SYSLIB DD  
//          DD DSN=MM01.TEST.OBJLIB, DISP=SHR  
//*-----*  
//GO.CUSTMAST DD DSN=MM01.CUSTMAST.DATA, DISP=SHR  
//GO.SALESRPT DD SYSOUT=*  
//GO.SYSOUT DD SYSOUT=*  
//
```

## JCL that executes a previously compiled and link-edited program

```
//MM01RN      JOB      (36512) , 'R MENENDEZ' , MSGLEVEL=(1,1) , REGION=4M,
//                               MSGCLASS=X, CLASS=A, NOTIFY=&SYSUID
//STEP1       EXEC     PGM=RPT1000
//STEPLIB     DD       DSN=MM01.TEST.LOADLIB, DISP=SHR
//CUSTMAST    DD       DSN=MM01.CUSTMAST.DATA, DISP=SHR
//SALESRPT    DD       SYSOUT=*
//SYSOUT      DD       SYSOUT=*
//
```

## How to execute a program using JCL

- Code an EXEC statement that identifies the program.
- Include a STEPLIB DD statement to identify the library that contains the load module for the program.
- If the program requires input or output files, code DD statements that identify those files.
- If a program accepts information from or displays information on a terminal, you should execute the program from TSO.

# Working with SDSF



## Basic skills for working with SDSF

- To start SDSF, enter SDSF at the TSO command prompt or select the SDSF option from the ISPF Primary Option Menu.
- To display any of the listed panels, enter the appropriate option in the command area and press Enter.

## Panel contents

- The *input queue* contains jobs that are waiting for execution and jobs that are currently executing.
- The *output queue* contains jobs that have completed execution and are waiting to be printed.
- The *held output queue* contains jobs that have completed execution and are held or assigned to a reserved class.
- The status panel displays information from all of the queues.

# The status panel

```
Mma - EXTRA! for Windows 98 / Windows NT
File Edit View Tools Session Options Help
Display Filter View Print Options Help
-----
SDSF STATUS DISPLAY ALL CLASSES                                LINE 1-10 (10)
COMMAND INPUT ==>                                           SCROLL ==> CSR
NP  JOBNAME  JOBID   OWNER   PRTY QUEUE      C  POS  SAFF  ASYS STATUS
    MM01     TSU04086 MM01     15 EXECUTION  DDC1 DDC1
    MM01CL   JOB03844 MM01      1 PRINT      C   85
    MM01CL   JOB06594 MM02      1 PRINT      A  256
    MM01CL   JOB06602 MM02      1 PRINT      C  257
    MM01CL   JOB00612 MM01      1 PRINT      A  486
    MM01CL   JOB00619 MM01      1 PRINT      A  492
    MM01RN   JOB01774 MM01      1 PRINT      A  571
    MM01CL   JOB02456 MM01      1 PRINT      A  661
    ? MM01CL   JOB05828 MM01      1 PRINT      A  891
    MM01RN   JOB04096 MM01      1 PRINT      A 1775
```

## How to work with job output using SDSF

- You can change some of a job's characteristics, such as its job class and priority, by typing over the appropriate fields.
- You can enter an action character in the NP column to handle job output in various ways.
- To print output in the held output queue, use the O action character to release the output and make it available for printing. If the output is assigned to a reserved class, you must also change the class so it's routed to the appropriate printer.

## Common action characters

Character	Function
S	Displays output data sets.
?	Displays a list of the output data sets for a job.
O	Releases output and makes it available for printing.
P	Purges output data sets.

# Using TSO to Execute Interactive Programs

## How to execute a program using TSO commands

- To execute a program from TSO, you use the CALL command to specify the name of the load module for the program.
- Before you can execute the program, you have to use ALLOCATE commands to allocate the data sets the program requires. With these commands, you relate the ddnames in the Select statements to the data sets that will be used.
- If a COBOL program uses Accept and Display statements for terminal I/O, you must allocate SYSIN and SYSOUT data sets to the terminal by coding an asterisk for the DSNNAME option.

## TSO commands for executing a COBOL program

```
ALLOCATE DDNAME(CUSTMAST) DSNAME(CUSTMAST.DATA)  
ALLOCATE DDNAME(SALESRPT) DSNAME(*)  
CALL TEST.LOADLIB(RPT1000)
```

## TSO commands for a program that includes Accept and Display statements

```
ALLOCATE DDNAME(SYSOUT) DSNAME(*)  
ALLOCATE DDNAME(SYSIN) DSNAME(*)  
CALL 'MM01.TEST.LOADLIB(CALC1000)'
```

# Lab 3

# Lab 1c

- **Compile, link and run a “Hello World” Cobol application.**

- Create/Run the program using JCL.
- Run the program using TSO

- **Step 1: Create 3 PDSs:**

- Create a data set for the cobol program:  
xxx.HELLO.COBOL
- Create a data set for the the JCL to compile/run the program: xxx.HELLO.JCL
- Create a data set for the resulting executable:  
xxx.HELLO.LOAD

*(where 'XXX' is your User Id)*



# Lab 3

- **Step 2:**

- Create the source code “Hello World” Cobol example:  
xxx.HELLO.COBOLO(FIRST)
- This would be done using ISPF. Note that the line numbers are created/updated automatically

# Lab 3: Cobol Program

The screenshot shows a COBOL editor window titled "Vista Session A". The menu bar includes File, Edit, Font, Transfer, Macro, Options, Window, and Help. The toolbar contains various icons for file operations, editing, and execution. The main text area displays the following COBOL code:

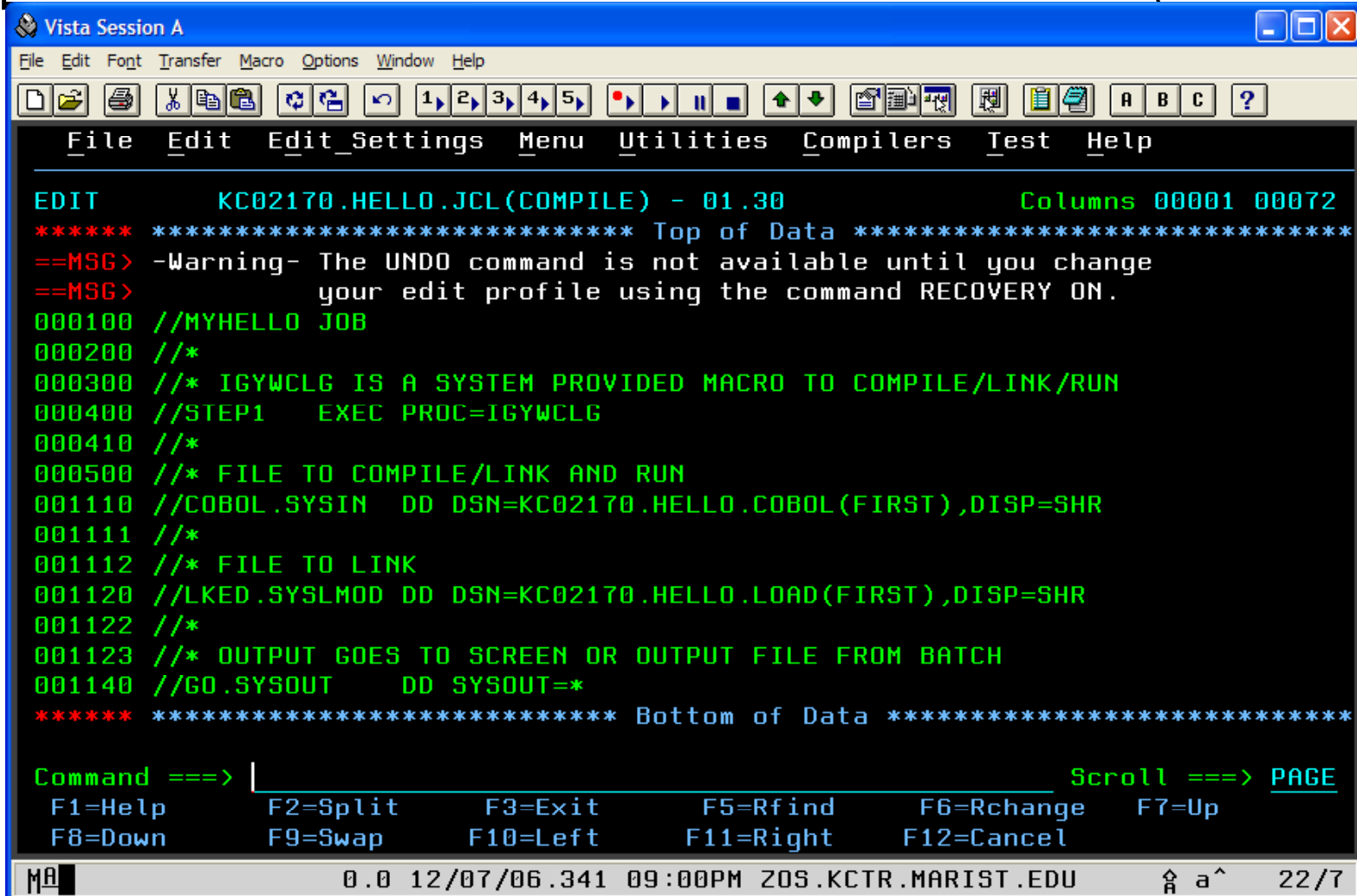
```
EDIT          KC02170.HELLO.COBO(LFIRST) - 01.14          Columns 00001 00072
==MSG>                your edit profile using the command RECOVERY ON.
000100 000100 Identification division.
000200 000200 Program-id. HELLO.
000210 000210 Environment division.
000230 000230 Data Division.
000270 000270 Procedure division.
000271 000271*
000272 000272* Hello world program
000273 000273*
000274 000274 MyFirstParagraph.
000280 000280      Display "Hello world".
000290 000290      stop run.
***** ***** Bottom of Data *****
```

At the bottom of the window, there is a status bar with the following information:

Command ==> Scroll ==> PAGE  
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up  
F8=Down F9=Swap F10=Left F11=Right F12=Cancel  
MA 0.0 12/07/06.341 08:56PM ZOS.KCTR.MARIST.EDU a 1,1

# Lab 3

- **STEP 3:** Create the JCL used to compile/link/run the hello world example. It should be in the JCL PDS: XXX.HELLO.JCL(COMPILE)



The screenshot shows a mainframe terminal window titled "Vista Session A". The menu bar includes File, Edit, Font, Transfer, Macro, Options, Window, and Help. The command bar contains various icons for file operations and editing. The main display area shows the following JCL code:

```
EDIT          KC02170.HELLO.JCL(COMPILE) - 01.30          Columns 00001 00072
***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000100 //MYHELLO JOB
000200 //*
000300 //* IGYWCLG IS A SYSTEM PROVIDED MACRO TO COMPILE/LINK/RUN
000400 //STEP1  EXEC PROC=IGYWCLG
000410 //*
000500 //* FILE TO COMPILE/LINK AND RUN
001110 //COBOL.SYSIN DD DSN=KC02170.HELLO.COBOL(FIRST),DISP=SHR
001111 //*
001112 //* FILE TO LINK
001120 //LKED.SYSLMOD DD DSN=KC02170.HELLO.LOAD(FIRST),DISP=SHR
001122 //*
001123 //* OUTPUT GOES TO SCREEN OR OUTPUT FILE FROM BATCH
001140 //GO.SYSOUT DD SYSOUT=*
***** Bottom of Data *****

Command ==> |
F1=Help      F2=Split    F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right    F12=Cancel
```

The status bar at the bottom shows the date and time as 0.0 12/07/06.341 09:00PM, the user as ZOS.KCTR.MARIST.EDU, and the page number as 22/7.

# Lab 3

- **STEP 4:** Execute the JCL – from within ISPF (when editing the JCL), enter the command ‘submit’ on the command line (aka the line that has “Command ==>” )
- **STEP 5:** View the results from executing the JCL. From the top level in ISPF:
  - Execute command 13 (SDSF).
  - Select O (to view output from the batch job).
  - You can Filter based on owner (your account id),
  - You should be able to see the output “MYHELLO” – that was the name of the JOB (you can see this in the JCL).
  - To view the output, one would enter an “?” in the left column prior to the name of the output. This will show the different “tasks in the JCL.
  - Then to see the actual output, enter a “s” next to the output you want to view (Hint: Look at the “GO” output)

# Lab 3

- **STEP 6:** Execute using TSO

(Using ISPF “command” - 6)

```
ALLOCATE DDNAME(SYSOUT) DSNAME(*)  
ALLOCATE DDNAME(SYSIN) DSNAME(*)  
CALL 'xxx.HELLO.LOAD(HELLO)'
```

## Common action characters for working with jobs

Character	Function
S	Displays output data sets.
?	Displays a list of the output data sets for a job.
A	Releases a held job.
C	Cancels a job.
H	Holds a job.
O	Releases held output and makes it available for printing.
P	Purges a job and its output.

## The output for a compile-and-link job

```
Mma - EXTRA! for Windows 98 / Windows NT
File Edit View Tools Session Options Help

Display Filter View Print Options Help
-----
SDSF OUTPUT DISPLAY MM01CL  JOB00612  DSID      2 LINE 0      COLUMNS 02- 81
COMMAND INPUT ==> █      SCROLL ==> CSR
***** TOP OF DATA *****
      J E S 2  J O B  L O G  --  S Y S T E M  D D C 1  --  N O D E

18.12.53 JOB00612 ---- MONDAY,    10 APR 2000 ----
18.12.53 JOB00612 IRR010I USERID MM01    IS ASSIGNED TO THIS JOB.
18.12.53 JOB00612 IEF677I WARNING MESSAGE(S) FOR JOB MM01CL  ISSUED
18.12.53 JOB00612 ICH70001I MM01    LAST ACCESS AT 18:10:20 ON MONDAY, APRIL 1
18.12.53 JOB00612 $HASP373 MM01CL  STARTED - INIT B    - CLASS A - SYS DDC1
18.12.53 JOB00612 IEF403I MM01CL - STARTED - TIME=18.12.53
18.12.56 JOB00612 -                                     --TIMINGS (M
18.12.56 JOB00612 -JOBNAME  STEPNAME PROCSTEP      RC  EXCP  CONN   TCB   SRB
18.12.56 JOB00612 -MM01CL  STEP1    COBOL          12   398   772   .00   .00
18.12.56 JOB00612 -MM01CL  STEP1    LKED      FLUSH      0     0   .00   .00
18.12.57 JOB00612 IEF404I MM01CL - ENDED - TIME=18.12.57
18.12.57 JOB00612 -MM01CL  ENDED.  NAME-A PRINCE          TOTAL TCB CPU TIM
18.12.57 JOB00612 $HASP395 MM01CL  ENDED

----- JES2 JOB STATISTICS -----
      10 APR 2000 JOB EXECUTION DATE
           12 CARDS READ
```

## How to work with job output

- You can use the standard ISPF scrolling commands to browse the output for a job.
- You can use the FIND and LOCATE commands to find a particular line of data in the output.
- If the output consists of more than one data set, you can use the NEXT and PREV commands to move to the next or previous data set.



## Job output that shows compile-time errors

```
Mma - EXTRA! for Windows 98 / Windows NT
File Edit View Tools Session Options Help

Display Filter View Print Options Help
-----
SDSF OUTPUT DISPLAY MM01CL JOB00612 DSID 101 LINE 102 COLUMNS 02- 81
COMMAND INPUT ==> SCROLL ==> CSR
PP 5648-A25 IBM COBOL for OS/390 & VM 2.1.1 CALC1000 Date 04
LineID Message code Message text

24 IGYPS2121-S "END-OF-SESSION-SWITCH" was not defined as a data-name. T
Same message on line: 35

36 IGYPS2011-E An "ELSE" did not have a matching "IF". The "ELSE" was di
37 IGYPS2121-S "TAX-AMOUNT" was not defined as a data-name. The statemen
Same message on line: 39

Messages Total Informational Warning Error Severe Terminating
Printed: 5
* Statistics for COBOL program CALC1000:
* Source records = 40
* Data Division statements = 3
* Procedure Division statements = 11
End of compilation 1, program CALC1000, highest severity 12.
```

# Types of compiler errors

Error type	Return code	Description
Informational (I)	0	Provides information only. The program will execute correctly without any modifications.
Warning (W)	4	Indicates a possible error. The program will most likely execute correctly without any modifications.
Error (E)	8	An error that the compiler has attempted to correct. The program will most likely require modification for it to execute correctly.
Severe (S)	12	A serious error that the compiler was unable to correct. The program will not execute correctly without modification.
Unrecoverable (U)	16	A serious error that caused the compilation to be terminated.

## How to handle compile-time errors

- If compile-time errors are detected on a mainframe, the job output will contain a description of each error, including the line number of the statement where the error was found and the message code for the error.
- The last character of the message code indicates the severity of the error.
- If the return code for any error is greater than 8, the link and go steps are cancelled.

## Job output that shows a run-time error

```
Mma - EXTRA! for Windows 98 / Windows NT
File Edit View Tools Session Options Help

Display Filter View Print Options Help
-----
SDSF OUTPUT DISPLAY MM01RN  JOB01774  DSID   101 LINE 1      COLUMNS 02- 81
COMMAND INPUT ==>                                SCROLL ==> CSR
DATE: 04/11/2000      YEAR-TO-DATE SALES REPORT      PAGE: 1
TIME: 12:15                                RPT1000
CUST              SALES              SALES
NUM  CUSTOMER NAME  THIS YTD      LAST YTD
11111 INFORMATION BUILDERS  1,234.56      1,111.11
12345 CAREER TRAINING CTR  12,345.67     22,222.22
CEE3207S The system detected a data exception (System Completion Code=0C7).
          From compile unit RPT1000 at entry point RPT1000 at statement 151 at co
          00006912.
***** BOTTOM OF DATA *****
```

## Common system completion codes

Code	Type	Description
0C1	Operation exception	Occurs when the system tries to perform an invalid operation like reading from or writing to an unopened file.
0C4	Protection exception	The program tried to access a storage area other than its own. Often happens when a table is accessed with an invalid index or subscript.
0C5	Addressing exception	Occurs when the system refers to a location in main storage that isn't available.
0C7	Data exception	Occurs when an operation is performed on a numeric field that has invalid data.
0CA	Decimal-overflow exception	Occurs when the result of an arithmetic operation can't be stored in a receiving field that's defined as a decimal number.
0CB	Decimal-divide exception	Occurs when the program tries to divide a number by zero.

## How to handle run-time errors

- If a run-time error occurs, the job output will contain a description of the error that includes a message code, a system completion code, and the line number of the statement that was executing when the error occurred.
- The system completion code tells you what type of error occurred, which helps you zero in on the cause of the error.